A Lecture Series on
DATA COMPRESSION

Abdou Youssef (Speaker)

Anastase Nakassis (MDVTG Manager)

INTRODUCTION

- What is Compression?
  - It is a process of deriving more efficient (i.e., smaller) representations of data

- Goal of Compression
  - Significant reduction in the data size to reduce the storage/bandwidth requirements

- Constraints on Compression
  - Perfect or near-perfect reconstruction (lossless/lossy)

- Strategies for Compression
  - Reducing redundancies
  - Exploiting the characteristics of human vision

NEED/MOTIVATION FOR COMPRESSION

- Massive Amounts of Data Involved in Storage/Transmission of Text, Sound, Images, and Videos in Many Applications
- Applications
  - Medical imaging
  - Teleradiology
  - Space/Satellite imaging
  - Multimedia
  - Video on demand
- Concrete Figures
  - A typical hospital generates close to 1 terabits per year
  - NASA's EOS will generate 1 terabytes per day
  - One 2-hour video = 1.3 terabits
  - Video transmission speed = 180Mb/sec
  - With MPEG1 (1.5Mb/s), need compression ratio of 120
  - With MPEG2 (4-10Mb/s), need comp. ratio of 18-45

BASIC DEFINITIONS

- Lossless Compression: 100% accurate reconstruction of the original data
- Lossy Compression: The reconstruction involves errors which may or may not be tolerable
- Bit Rate: Average number of bits per original data element after compression
- Compression Ratio: $\frac{Original\ Data\ Size}{Compressed\ Data\ Size}$
- Coding: Compression
- Codeword: A binary string representing either the whole coded data or one coded data symbol

STRATEGIES FOR COMPRESSION
(Redundancy Reduction)

- Symbol-Level Representation Redundancy
  - Different symbols occur with different frequencies
  - Variable-length codes vs. fixed-length codes
  - Frequent symbols are better coded with short codes
  - Infrequent symbols are coded with long codes
  - Example Techniques: Huffman Coding

- Block-Level Representation Redundancy
  - Different blocks of data occur with varying frequencies
  - Better then to code blocks than individual symbols
  - The block size can be fixed or variable
  - The block-code size can be fixed or variable
  - Frequent blocks are better coded with short codes
  - Example techniques: Block-oriented Huffman, Run-Length Encoding (RLE), Arithmetic Coding, Lempil-Ziv (LZ)

REDUNDANCY REDUCTION (Cont.)

- Inter-Pixel Spatial Redundancy
  - Neighboring pixels tend to have similar values
  - Neighboring pixels tend to exhibit high correlations
  - Techniques: Decorrelation and/or processing in the frequency domain
  - Spatial decorrelation converts correlations into symbol- or block-redundancy
  - Frequency domain processing addresses visual redundancy (see the next slide)

- Inter-Pixel Temporal Redundancy (in Video)
  - Often, the majority of corresponding pixels in successive video-frames are identical over long spans of frames
  - Due to motion, blocks of pixels change in position but not in values between successive frames
  - Thus, block-oriented motion-compensated redundancy reduction techniques are used for video compression

REDUNDANCY REDUCTION (Cont.)

- Visual Redundancy
  - The human visual system (HVS) has certain limitations that make many image contents invisible. Those contents, termed visually redundant, are the target of removal in lossy compression.
  - In fact, the HVS can see within a small range of spatial frequencies: 1–60 cycles/arc-degree
  - Approach for reducing visual redundancy in lossy compression
    1. Transform: Convert the data to the frequency domain
    2. Quantize: Under-represent the high frequencies
    3. Losslessly compress the quantized data

INFORMATION THEORY PRELIMINARIES

- Discrete Memoryless Source S: A data generator where the alphabet $\{a_k\}$ is finite and the symbols generated are independent of one another.

- Entropy: H(S)=-$\Sigma_k \, p_k \log p_k$, where p_k =Prob$[a_k]$

- H(S) is the minimum average number of bits/symbol possible

- Sources with Memory: Presence of inter-symbol correlation

- Their entropy is still the min average number of bit/symbol

- Adjoint Source of Order N
  - Treat each possible block A of N symbols as a macrosymbol, and compute the probability P_A
  - Treat the source as a memoryless source consisting of the macrosymbols A's and their probabilities P_A's
  - The entropy $H_N = -\,\Sigma_A \, P_A \log P_A$

- Theorem (Shannon): For any source S with memory, $\frac{H_N(S)}{N} \longrightarrow H(S)$ as $N \longrightarrow \infty$

HUFFMAN CODING

- Each (macro)symbol A has a probability P_A
- Form a Huffman tree as follows:
  1. Create a node for each symbol
  2. While (there are two or more uncombined nodes) do
     - select 2 uncombined nodes a & b of minimum probabilities
     - Combine a & b by creating a new node c of prob P_a+P_b, and making a & b children of c
  3. Label the tree edges: left edges with 0, right edges with 1
  4. The code of each symbol is the binary sequence labeling the path from the root down to the corresponding leaf

HUFFMAN CODING (Cont.)

- Example: Alphabet=$\{A, B, C, D, E, F, G, H\}$ of probabilities $1/2, 1/4, 1/16, 1/16, 1/32, 1/32, 1/32, 1/32$

HUFFMAN CODING (Cont.)

- Coding a Sequence/File
  - Represent each symbol in the sequence by its Huffman code
  - Example: $ABBAACA$ is coded as 101011100101
- Decoding
  - Proceed from the next undecoded bit, and walk down the tree (starting from the root) going left or right depending on whether the bit is 0 or 1
  - When a leaf is reached, replace the binary string just scanned by the symbol corresponding to the leaf
  - Example:

RUN-LENGTH ENCODING

- Represent each subsequence of identical symbols by a pair (L, a) where L is the length of the subsequence, and a is the recurring symbol in the subsequence

- Example: $aaabbbbaaaa$ is coded as $(3, a)$ $(4, b)$ $(4, a)$

- If the sequence is binary, there is no need to represent a because the value of a alternates between 0 and 1

- Example: 00011111000011 is coded as $3, 5, 4, 2$

- RLE can be followed by Hoffman coding to further code the L's and a's

- The fax standard uses RLE

ARITHMETIC CODING

- Arithmetic Coding achieves a bit rate equal to the entropy
- It codes the whole input sequence, rather than individual symbols, into one codeword
- The Conceptual Main Idea
  - For each binary input sequence of n bits, divide the unit interval into 2^n intervals, where the length of i-th interval I_i is the probability of the i-th n-bit binary sequence
  - Code the i-th binary sequence by $l_1 l_2 ... l_t$ where $0.l_1 l_2 ... l_t ...$ is the binary representation of the left end of interval I_i, and $t = \lceil -\log(\text{Prob}(i\text{-th sequence})) \rceil$

ARITHMETIC CODING

- Method
  1. Let $a_1a_2...a_n$ be the input to be coded
  2. Let $I = [L, R)$ be the interval corresponding to the subsequence scanned so far
  3. Initially, $I = [0, 1)$;
  4. for $i = 1$ to n do
     - Let $P_i = Prob[0/a_1a_2...a_{i-1}]$, and $\Delta = R - L$
     - Divide I into 2 intervals: $[L, L+P_i\Delta)$ and $[L+P_i\Delta, R)$
     - If $a_i = 0$, reduce I to $[L, L + P_i\Delta)$
     - Else, reduce I to $[L = L + P_i\Delta, R)$
  5. $t = \lceil -\log(R - L) \rceil$
  6. Express L in binary $L = 0.l_1l_2...$
  7. Code the input with $l_1l_2...l_t$

- Patent: IBM Q-Coder

ARITHMETIC CODING (Cont.)

- Example: Binary Markov Source $P[0/0] = P[1/1] = 3/4$, $P[0/1] = P[1/0] = 1/4$ and $P[0] = P[1] = 1/2$

LEMPIL-ZIV COMPRESSION

- LZ encodes recurring patterns (blocks) using the positions of their first occurrences

- LZ Encoder

  1. Let $x_1 x_2 ... x_n$ be the input to be coded
  2. Maintain a dictionary (DICT) of patterns seen so far
  3. DICT[1]=x_1, and put x_1 in the output code
  4. While (there are still input symbols) do
     - Read from the remaining input until the string scanned is no longer in DICT. Call that string Wa, where W is in DICT and a in the input symbol after W
     - Let j be the index where W=DICT[j]; ($j < i$)
     - Let $\overline{j}$ be the $\lceil \log i \rceil$-bit binary representation of j
     - Code Wa as $(\overline{j}, a)$ and append that code to the output
     - DICT[i] $= Wa$ and $i = i + 1$

- Remark: The dictionary is not stored/transmitted.

- The LZ bitrate is asymptotically optimal <u>without</u> the need to know or compute the underlying probability model of the input data.

LEMPEL-ZIV (Cont.)

- Example: $x = 0010100100010010100110101$

LEMPEL-ZIV (Cont.)

- LZ Decoder

  1. Let $y = y_1y_2...$ be the codeword to be decoded back to x
  2. $x = y_1$ and DICT[1] $= y_1$
  3. $i = 2$
  4. While (the codeword is not fully scanned) do
     - $j =$ the next $\lceil \log i \rceil$ bits from y
     - W=DICT[j]
     - a= the next symbol from y
     - append Wa to the right of x
     - DICT[i] $= Wa$
     - $i = i + 1$

LEMPEL-ZIV (Cont.)

- Example: Decoding $y = 01110011010011011011 1$ from the previous example

i	$\lceil \log i \rceil$	$\bar{j} = j$	W	a	DICT[i]	$x = (\text{previous}(x))(Wa)$
1	0	$\epsilon = \epsilon$	ϵ	0	0	0
2	1	$(1)_2$=1	0	1	01	001
3	2	$(10)_2 = 2$	01	0	010	001010
4	2	$(11)_2 = 3$	010	0	0100	0010100100
5	3	$(100)_2 = 4$	0100	1	01001	001010010001001
6	3	$(101)_2 = 5$	01001	1	010011	001010010001001010011
7	3	$(011)_2 = 3$	010	1	0101	0010100100010010100110101

- Below, the underbraced strings in y are the binary representations of the various values of j

- Right under each underbraced j value, the corresponding DICT[j] is put in x. The non-underbraced bits of y are "dropped" into the appropriate positions in x.

$$
\begin{array}{lllllll}
y = \underbrace{0} & \underbrace{1}\;1 & \underbrace{10}\;0 & \underbrace{11}\;0 & \underbrace{100}\;1 & \underbrace{101}\;1 & \underbrace{011}\;1 \\
x = 0 & 0\;\;1 & 01\;0 & 010\,0 & 0100\,1 & 010011 & 010\;1
\end{array}
$$

DPCM

- DPCM is a predictive technique that capitalizes on inter-pixel spatial redundancy
- DPCM predicts the next pixel based on the values of the previous neighboring pixels
- It then computes the residual pixel (actual − predicted)
- Finally, it losslessly compresses the residual data, using RLE, Huffman, etc.
- DPCM decorrelates the data and causes the residual to have lower (memoryless) entropy
- DPCM is the lossless JPEG standard

PROS/CONS OF THE LOSSLESS TECHNIQUES

	Advantages	Disadvantages
Huffman	• Easy to implement • Good bitrate	• Ignores correlations
Blocked-Huffman	• Exploits correlations • Near-optimal bitrate	• Block probabilities are costly to compute
Arithmetic Coding	• Optimal bitrate	• Precision problems as intervals become very small • Needs the conditional probability model
RLE	• Easy to implement • Good bitrate	• Not generally applicable as a standalone
LZ	• Optimal bitrate • Does not need the probability model of the data	• Requires long input sequences to pay off
DPCM	• Easy to implement • Good bitrate	• Limited to inter-pixel redundancy
Other predictive coders	• Good bitrate	• Slower than DPCM
Bit-Plane Coding	• Good bitrate	• Slower than DPCM

LIMITATIONS OF LOSSLESS COMPRESSION

- Low compression ratios (about 2 to 1)
- No lossless compression technique can compress every possible input by at least one bit